In the meantime, it is an appealing idea that representation and learning techniques from both symbolic processing models and connectionist network models shall be brought together to tackle problems that neither type of model alone can apparently handle very well.

One such problem is the modeling of human cognition, which requires dealing with a variety of cognitive capacities. Several researchers (e.g., Smolensky 1988, Dreyfus and Dreyfus 1986) have consistently argued that cognition is multifaceted and better captured with a combination of symbolic and connectionist processes. Many methods and frameworks reviewed above share the belief that connectionist and symbolic methods can be usefully integrated, and such integration may lead to significant advances in our understanding of cognition. It thus appears that hybridization is a theoretically sound approach, in addition to being a practically expedient approach.

*See also*: Artificial and Natural Computation; Artificial Intelligence: Connectionist and Symbolic Approaches; Artificial Intelligence in Cognitive Science; Artificial Neural Networks: Neurocomputation; Connectionist Approaches; Connectionist Models of Concept Learning; Connectionist Models of Language Processing; Intelligence: History of the Concept; Knowledge Representation; Scientific Discovery, Computational Models of

## *Bibliography*


Barnden J A, Pollack J B (eds.) 1991 *Advances in Connectionist and Neural Computation Theory*. Ablex Publishing Co., Norwood, NJ

Davis E 1990 *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, CA

Dreyfus H L, Dreyfus S E 1986 *Mind Over Machine*. Free Press, New York

Giles C L, Gori M 1998 *Adaptive Processing of Sequences and Data Structures*. Springer, New York

Lange T, Dyer M 1989 High-level inferencing in a connectionist network. *Connection Science* **1**: 181–217

McCarthy J L 1968 Programs with common sense. In: Minsky M (ed.) *Semantic Information Processing*. MIT Press, Cambridge, MA

Rumelhart D E, McClelland J L, PDP Research Group 1986 *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*. MIT Press, Cambridge, MA

Michalski R S 1983 A theory and methodology of inductive learning. *Artificial Intelligence* **20**: 111–61

Miikkulainen R, Dyer M 1991 Natural language processing with modular PDP networks and distributed lexicons. *Cognitive Science* **15**(3): 343–99

Newell A, Simon H 1976 Computer science as empirical inquiry: Symbols and search. *Communication of ACM* **19**: 113–26

Pearl J 1988 *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA

Pollack J B 1990 Recursive distributed representations. *Artificial Intelligence* **46**(1–2): 77–106

Quillian M 1968 Semantic memory. In: Minsky M (ed.) *Semantic Information Processing*. MIT Press, Cambridge, MA, pp. 216–60

Quinlan R 1986 Inductive learning of decision trees. *Machine Learning* **1**: 81–106

Russell S J, Norvig P 1995 *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ

Shavlik J W, Dietterich T G 1990 *Reading in Machine Learning*. Morgan Kaufmann, San Mateo, CA

Shastri L, Ajjanagadde V 1993 From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences* **16**(3): 417–51

Smolensky P 1988 On the proper treatment of connectionism. *Behavioral and Brain Sciences* **11**(1): 1–23

Sun R 1994 *Integrating Rules and Connectionism for Robust Commonsense Reasoning*. Wiley, New York

Sun R, Alexandre F (eds.) 1997 *Connectionist Symbolic Integration*. Erlbaum, Mahwah, NJ

Sun R, Bookman L A (eds.) 1995 *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer, Boston

Sun R, Peterson T 1998 Autonomous learning of sequential tasks: Experiments and analyzes. *IEEE Transactions on Neural Networks* **9**(6): 1217–34

Turing A 1950 Computing machinery and intelligence. *Mind* **59**: 433–60

Waltz D, Feldman J A (eds.) 1988 *Connectionist Models and Their Implications*. Ablex, Norwood, NJ

Zadeh L A 1988 Fuzzy logic. *Computer* **21**(4): 83–93


R. Sun

# Artificial Intelligence: Genetic Programming

## *1. Introduction*

The term genetic programming (GP) describes a research area within artificial intelligence (AI) that deals with the evolution of computer code. The term evolution refers to an artificial process analogous to natural evolution of living organisms, but which has been abstracted and stripped of most of its intricate details. The resultant algorithms then yield approximate solutions to problems in machine learning or induce precise solutions in the form of grammatically correct (language) structures for the automatic programming of computers.

Genetic programming is part of the growing set of evolutionary algorithms which apply search principles analogous to those of natural evolution in a variety of different problem domains, notably parameter optimization. Evolutionary programming, evolutionary strategies, and genetic algorithms are three other

VISIT…

LANZAROTE Caliente.COM

branches of the area of evolutionary algorithms which mostly find applications as optimization techniques.

All evolutionary algorithms follow Darwin's principle of differential natural selection. This principle states that the following preconditions must be fulfilled for evolution to occur via (natural) selection:

There are entities called individuals which form a population. These entities can reproduce or can be reproduced.

There is heredity in reproduction, that is to say that individuals produce similar offspring.

In the course of reproduction there is variety which affects the likelihood of survival and therefore of reproducibility of individuals.

There are finite resources which cause the individuals to compete. Due to overreproduction of individuals not all can survive the struggle for existence. Differential natural selection exerts a continuous pressure towards improved individuals.

The representation of programs, or generally structures, in GP has a strong influence on the behavior and efficiency of the resulting algorithm. As a consequence, many different approaches toward choosing representations have been adopted in GP. The resulting principles have been applied even to other problem domains such as design of electronic circuits or art and musical composition.

## 2. *The Mechanisms Behind Genetic Programming*

Genetic programming works with a population of programs that are executed or interpreted in order to judge their behavior. Usually, a scoring operation called fitness measurement is applied to the outcome of the behavior. For instance, the deviation between the quantitative output of a program and its target value (defined through an error function) could be used to judge the behavior of the program. This is straightforward if the function of the target program can be clearly defined. Results may also be defined as side-effects of a program, such as consequences of the physical behavior of a robot controlled by a genetically developed program. Sometimes, an explicit fitness measure is missing, for instance in a game situation, and the results of the game (winning or losing) are taken to be sufficient scoring for the program's strategy. The general approach is to apply a variety of programs to the same problem and to compare their performance relative to each other (see Fig. 1).

The outcomes of fitness measurement are used to select programs. There are a number of different

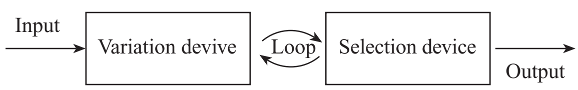


***Figure 1***
The variation selection loop of GP

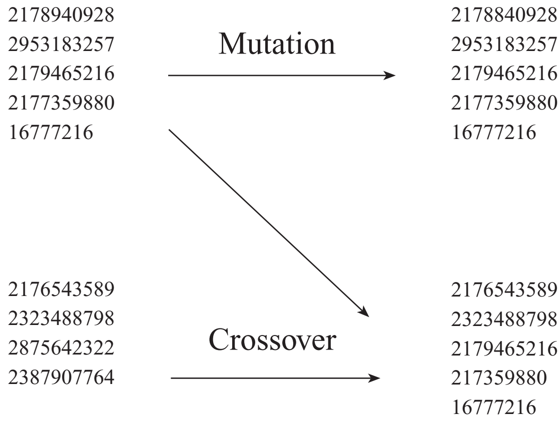


***Figure 2***
The primary operations of GP, mutation, and crossover are applied here to programs represented by the sequences of instructions. The instructions are coded as integer numbers which allows easy manipulation by access to these numbers

methods for selection, both deterministic and stochastic. Selection determines (a) which programs are allowed to survive (overproduction selection), and (b) which programs are allowed to reproduce (mating selection). Once a set of programs has been selected for further reproduction, the following operators are applied: reproduction, mutation, and crossover. Reproduction simply copies an individual, mutation varies the structure of an individual under control of a random number generator, and crossover mixes the structure of two (or more) programs to generate one or more new programs (see Fig. 2). Additional variation operators are applied in different applications. Most of these contain problem knowledge in the form of heuristic search recipes which are adapted to the problem domain.

In this way, fitness advantages of individual programs are exploited in a population to lead to better solutions. A key effort in GP is the definition of the fitness measure. Sometimes the fitness measure has to be iteratively improved in order for the evolved solutions actually to perform the function they were intended for. The entire process can be seen in close analogy to breeding animals. The breeder has to select those individuals from the population which carry the targeted traits to a higher degree than others.

Researchers (Turing 1950, Friedberg 1958, Samuel 1959) suggested ideas similar to GP already in the early days of AI, but did not get very far. So it was only after other techniques in evolutionary algorithms had been successfully developed that GP emerged. Earlier work concerned genetic algorithms (Holland 1975), evolution strategies (Schwefel 1981), and evolutionary programming (Fogel et al. 1966). These methods have been applied successfully to a wide spectrum of

problem domains, especially in optimization. However, it was unclear for a long time whether the principles of evolution could be applied to computer code, with all its dependencies and structural brittleness. Negative results from early experiments seemed to indicate that evolution of computer code was not possible. Successes were all in the area of constraint optimization (Michalewicz 1996), where methods were made available for how to deal with structural brittleness. These methods found their way into programming and gave rise to the new field of GP (Koza 1992).

## 3. *Progress and State-of-the-art*

In his seminal work of 1992, Koza established the field of GP by arguing convincingly that manipulation of symbolic tree structures is possible with evolutionary algorithms and that the resulting technique would have a wide variety of applications. In subsequent years, the field experienced both broadening and deepening (Banzhaf et al. 1998). Many different representations for GP were studied, among them other generic data structures such as sequences of instructions or directed graphs, as well as more exotic data structures such as stacks or neural networks. Today, different approaches are considered as GP, from the evolution of parse trees to the evolution of arbitrary structures. The overarching principle is to subject structures with variable complexity to forces of evolution by applying mutation, crossover, and fitness-based selection. The results must not necessarily be programs.

An ever-present difficulty with GP is that the evolution of structures of variable complexity leads often to large structures with considerable redundancy. Notably, variable complexity often leads to inefficient and space-consuming code. It was subsequently recognized that the evolutionary forces exerted a pressure toward more complex solutions, most of which could be removed after evolution without doing any harm to the evolved solution. By drawing an analogy from biological evolution of genomes, this phenomenon was called 'intron growth,' or growth of ineffective code. Thought the explanation for this phenomenon is not fully understood yet, it was found that at least two different influences were at work promoting the growth of complexity during evolution. The most important one has to do with the protection effect of redundant code if subjected to the action of crossover or mutation. Redundant code was resistant to crossover and mutation and allowed its carrier solution to survive better, compared to other individuals which did not possess the redundancy.

Currently, many researchers are working to transfer results from research in genetic algorithms to genetic programming. To achieve results in GP is generally more difficult since it works with variable complexity and multiple fitness cases for fitness scoring. The schema theory of genetic algorithms (Goldberg 1989, Vose 1999) has been a primary target of knowledge transfer. In the meantime, a number of different schema theorems have been formulated for GP.

When analyzing search spaces of programs it was realized that their size is many orders of magnitude larger than search spaces of combinatorial optimization problems. A typical size for a program search space might be $10^{100,000}$, as opposed to a typical search space for a combinatorial optimization problem of the order of $10^{100}$. Although this might be interpreted as discouraging for search mechanisms, it was also realized that the solution density in program spaces is, above a certain threshold, constant with changing complexity (Langdon 1999). In other words, there are proportionally many more valid solutions in program spaces than in the spaces of combinatiorial optimization problems.

## 4. *Applications*

The main application areas of GP are (from narrow to wide) (Banzhaf et al. 1998): computer science, science, engineering, and art and entertainment. In computer science, the development of algorithms has been a focus of attention. By being able to manipulate symbolic structures, genetic programming is one of the few heuristic search methods for algorithms. Sorting algorithms, caching algorithms, random number generators, and algorithms for automatic parallelization of code (Ryan 2000), to name a few, have been studied. The spectrum of applications in computer science spans from the generation of proofs for predicate calculus to the evolution of machine code for accelerating function evaluation. The general tendency is to try to automate the design process for algorithms of different kinds.

Typical applications in science are to modeling and pattern recognition. Modeling certain processes in physics and chemistry with the unconventional help of evolutionary creativity supports research and understanding of the systems under study. Pattern recognition is a key ability in molecular biology and other branches of biology, as well as in science in general. Here, GP has delivered first results that are competitive if not better than human-generated results.

In engineering, GP is used in competition or cooperation with other heuristic methods such as neural networks or fuzzy systems. The general goal is again to model processes such as production plants, or to classify results of production. Control of man-made apparatus is another area where GP has been used successfully, with process control and robot control the primary applications.

In art and entertainment, GP is used to evolve realistic animation scenes and appealing visual graphics. It also has been used to extract structural in-

formation from musical composition in order to model the process so that automatic composition of music pieces becomes possible.

Many of these problems require a huge amount of computational power on the part of the GP systems. Parallel evolution has hence been a key engineering aspect of developments in GP. As a paradigm, genetic programming is very well suited for a natural way of parallelization. With the advent of inexpensive parallel hard- and software (Sterling et al. 1999), a considerable proliferation of results is expected from GP systems.

### 5. *Methodological Issues and Future Directions*

In recent years, some researchers have claimed human-competitive results in the application of genetic programming to certain problems (Koza et al. 1999). These claims are based on a comparison between the best known at present human solution to a problem and its GP counterpart. Usually, a large amount of computational power had to be invested in order to gain human-competitive results from genetic programming runs. Due to the ability of the human mind quickly to grasp the recipes of problem solution an artificial system has applied, the question remains open whether GP solutions will stay better than human solutions.

Theory of genetic programming is presently greatly underdeveloped and will need to progress quickly in order to catch up with other evolutionary algorithm paradigms. Most of the obstacles stem from the fact of variable complexity of solutions evolved in GP. Implementation of GP will benefit in the coming years from new approaches which include research from developmental biology. Also, it will be necessary to learn to handle the redundancy forming pressures in the evolution of code.

Application of genetic programming will continue to broaden. Many applications focus on controlling behavior of real or virtual agents. In this role, genetic programming may contribute considerably to the growing field of social and behavioral simulations. Genetic algorithms have already been found beneficial in optimizing strategies of social agents (Gilbert and Troitzsch 1999). With its ability to adjust the complexity of a strategy to the environment and to allow competition between agents, GP is well positioned to play an important role in the study and simulation of societies and their evolution.

*See also*: Adaptation, Fitness, and Evolution; Artificial Intelligence: Connectionist and Symbolic Approaches; Artificial Intelligence: Search; Evolution: Optimization; Evolution: Self-organization Theory; Evolutionary Theory, Structure of; Intelligence: History of the Concept; Natural Selection

### *Bibliography*


Banzhaf W, Nordin P, Keller R, Francone F 1998 *Genetic Programming—An Introduction on the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Francisco, CA

Fogel L J, Owens A, Walsh M 1966 *Artificial Intelligence Through Simulated Evolution*. Wiley, New York

Friedberg R M 1958 A learning machine: part I. *IBM Journal of Research and Development* **2**: 2–13

Gilbert N G, Troitzsch K G 1999 *Simulation for the Social Scientist*. Open University Press, Buckingham, UK

Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA

Holland J H 1975 *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI

Koza J R 1992 *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA

Koza J R, Andre D, Bennett F, Keane M 1999 *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann, San Francisco, CA

Langdon W 1999 Boolean function fitness spaces. In: Poli R, Nordin P, Langdon W, Fogarty T (eds.) *Proceedings EuroGP'99*. Springer, Berlin, pp. 1–14

Michalewicz Z 1996 *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin

Ryan C 2000 *Automatic Re-engineering of Software Using Genetic Programming*. Kluwer, Boston, MA

Samuel A 1959 Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* **3**: 210–29

Schwefel H P 1981 *Numerical Optimization of Computer Models*. Wiley, Chichester, UK

Sterling T L, Salmon J, Becker D, Savarese D 1999 *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*. MIT Press, Cambridge, MA

Turing A M 1950 Computing machinery and intelligence. *Mind* **59**: 433–60

Vose M D 1999 *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA


W. Banzhaf

## Artificial Intelligence in Cognitive Science

Artificial intelligence (AI) and cognitive science are, at the turn of the twenty-first century, two distinct disciplines, with overlapping methodologies, but rather different goals. AI is a branch of computer science and is concerned with the construction and deployment of intelligent agents as computer programs, and also with understanding the behavior of these artifacts. The core scientific goal of AI is to understand basic principles of intelligent behavior that apply equally to animal and artificial systems (Russell and Norvig 2000). Almost all of the work is math-




International Encyclopedia of the Social & Behavioral Sciences ISBN: 0-08-043076-7